Semantic space of logic

Orges Leka

27. Mai 2023 in Limburg

Inhaltsverzeichnis

1 Introduction and aim **1**

2 Lukasiewicz logic and Liar paradoxon **3**

3 One logic and semantic of natural numbers **5**

4 Simplex logic and semantics, Boolean logic **6**

5 Further ideas **8**

6 Appendix, Sagemath / Python code **8**

1 Introduction and aim

We want to use the theory of the reproducing kernel Hilbert spaces as a start for a semantic geometry of logic. In order for these phrases to have a meaning in the 'real world' we want to give some examples:

If we start with a set of things such as words, objects of the real world, objects of the abstract world, natural numbers etc. to make it short, let me start with a set X and a positive semidefinite kernel k on this set such that $-1 \leq k(x, y) \leq 1$ and $k(x, x) = 1$ for all $x, y \in X$. In machine learning the so called 'kernel trick' is really a trick which helps us do computations in an infinite space with only a finite amount of symbols and resources, which kind of is a miracle, if one thinks about it. It has been usefully used with support vector machines to solve real world problems of 'classification'. In the background there is of course geometry. To the best of my search engine knowledge David Miller and Jonathan Westphal were the first to take the geometry of logic literally in their work 'A geometry of logic' and 'Logic of vectors'. Here we want to use vectors to measury similarity between two things x, y which might be axioms, propositions or formulas or even words and sentences in a precise mathematical way. Using this formalism of logic paired with geometry in a semantic way, one can give 'meaning' to the Liar paradox, find the Boolean logic and Lukasiewicz logic embedded in this geometry so to say.

The ideas here are to first start with a set of things and a positive semidefinite kernel on these things. The theory of RKHS / Moore-Aaronszajn theorem, gives us a Hilbert space H where miracously $k(x,y) = \langle \phi(x), \phi(y) \rangle$ where $\phi : X \to H$ is an injective function, called 'feature mapping' in machine learning. Now suddenly we can do geometry on this set abstract set of things X. The conclusion of machine learning community is, that a 'feature vector' $\phi(x)$ of x captures the 'semantic of the element x' relative to k.

Taking the perspective of everyday logic, we might want be able to give a semantic meaning to the phrase 'different point of views'. But this is very easy done in geometry, since if x takes the perspective of w, then we might want to project the unit vector $\phi(x)$ to the unit vector $\phi(w)$, to get:

$$\pi_w(x) = \frac{k(w,x)}{|w|^2} w = k(w,x)w \tag{1}$$

For this last equation, we write:

$$x \text{ (relative to } w) := \pi_w(x) = k(w,x)w \tag{2}$$

and

$$|x|_w := |\pi_w(x)| \tag{3}$$

which is the length of the projected vector.

Now we can imagine what the meaning of 'I have a different perspective' means. We simply associate the projection of the semantic vector of x to some other perspective vector w. Then 'changing perspective' simply means to change the perspective vector from w to say $\hat{w}$.

Let us define

$$x \wedge y \text{ (relative to } w) := \min(k(w,x), k(w,y))\phi(w) \tag{4}$$

$$x \vee y \text{ (relative to } w) := \max(k(w,x), k(w,y))\phi(w) \tag{5}$$

$$\neg x := -\pi_w(x) \tag{6}$$

and following Lukasiewicz:

$$x \to y \text{ (relative to } w) := \min(1, 1 + |y|_w - |x|_w)w \tag{7}$$

$$x \leftrightarrow y \text{ (relative to } w) := x \equiv y := (x \to y) \wedge (y \to x) \tag{8}$$

We have the de Morgan rules, double negation, contraposition:

$$(\neg x) \wedge (\neg y) = \neg(x \vee y) \text{ (relative to } w) \tag{9}$$

$$(\neg x) \vee (\neg y) = \neg(x \wedge y) \text{ (relative to } w) \tag{10}$$

$$(\neg(\neg x)) = x \text{ (relative to w)} \tag{11}$$

$$(x \to y) = ((\neg y) \to (\neg x)) \text{ (relative to w)} \tag{12}$$

Let us define three cases of truth values: T, F, I which should be translated to True, False, Indeterminate.

$$\mu(x) = \begin{cases} T & \text{if } k(w,x) < 0, \\ I & \text{if } k(w,x) = 0, \\ F & \text{if } k(w,x) > 0. \end{cases} \tag{13}$$

We have a version of 'Modus ponens':

$$\text{If } \mu(x) = T \text{ and } \mu(x \to y) = T \text{ then } \mu((x \wedge (x \to y)) \to y)) = T \tag{14}$$

2 Lukasiewicz logic and Liar paradoxon

Let me give a concrete example:

$X := \{-e_1, e_0, e_1\}$ where $e_1 = (1,0), e_0 = (0,1), -e_1 = (-1,0)$ are vectors in $\mathbb{R}^2$ with the usual dot product.

Gram matrix:

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \tag{15}$$

Now we will loop with w through all three different elements of X to get different perspectives:

$$w = (0,\ 1) \tag{16}$$

Not

0 0
1 -1
0 0

And

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Or

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Implies

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

Iff

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

Perspecitve change:

$$w = (1, 0)$$

Not

-1 1
0 0
1 -1

And

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

Or

$$\begin{pmatrix} -1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Implies

$$\begin{pmatrix} 1 & 0 & -1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Iff

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

Perspecitve change:

$$w = (-1, 0)$$

Not

1 -1
0 0
-1 1

And

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Or

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & -1 \end{pmatrix}$$

Implies

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

Iff

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

Interpretation: If we associate $T := e_1, F := -e_1, I := e_0, w = e_1$ then we get the following interpretation of the Liar paradox:

$$\neg e_0 = e_0$$

This is a sentence which from our current perspective ($w = e_1$) contradicts itself.

Changing the perspective to $w = e_0$ we get:

$$e_0 = T$$

but then the previous (from the perspective before) vectors $-e_1, e_1$ both get the value I and are indeterminate. They are equivalent in the sense defined above, although as vectors not equal. So from this perspective the vectors $-e_1, e_1$ 'lose' their previous meaning of T, F to become equivalent $I \equiv I$.

In conclusion we can say, that the Liar Paradox can be given a semantic which depends on the perspective, without losing some formal symbolism of logic.

3 One logic and semantic of natural numbers

The function $k(a,b) = \frac{\gcd(a,b)^2}{ab}$ is a p.d. kernel similarity on the natural numbers. By the theory of Reproducing Kernel Hilbert space, there exists a Hilbert space H and an embedding of the natural numbers $\phi : \mathbf{N} \to H$ such that:

$$k(a,b) = \langle \phi(a), \phi(b) \rangle$$

In practice, given finitely many natural number $a_1, \cdots, a_r$ we can compute such an embedding for those numbers using the Cholesky decomposition of the Gram matrix $G = (k(a_i, a_j))_{i,j} = CC^T$ and then assigning to a_i the i-th column vector of C.

In the given specific case however, we can give a direct embedding for all numbers:

Let e_d be the d-th standard-basis vector in the Hilbert space $H = l_2(\mathbb{N})$. Let $h(n) = J_2(n)$ be the second Jordan totient function. Define:

$$\phi(n) = \frac{1}{n} \sum_{d|n} \sqrt{h(d)} e_d$$

.

Then we have:

$$\langle \phi(a), \phi(b) \rangle = \frac{\gcd(a,b)^2}{ab} =: k(a,b)$$

The vectors $\phi(a_i)$ are linearly independent for each finite set $a_1, \cdots, a_n$ of natural numbers, since

$$\det(G_n) = \prod_{i=1}^{n} \frac{h(a_i)}{a_i^2}$$

is not zero, where G_n denotes the Gram matrix.

4 Simplex logic and semantics, Boolean logic

Let $X := \{\pm e_n | 1 \leq n \leq N\}$ N be a natural number.

Gram matrix:

$$\begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{pmatrix}$$

$$w = \left(\sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}} \right)$$

Not

$$sqrt(1/3) - sqrt(1/3)$$

$$sqrt(1/3) - sqrt(1/3)$$

$$sqrt(1/3) - sqrt(1/3)$$

$$-sqrt(1/3)sqrt(1/3)$$

$$-sqrt(1/3)sqrt(1/3)$$

$$-sqrt(1/3)sqrt(1/3)$$

And

$$\begin{pmatrix} \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} \\ \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} \\ \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} \\ -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} \\ -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} \\ -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} \end{pmatrix}$$

Or

$$\begin{pmatrix} \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} \\ \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} \\ \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} \\ \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} \\ \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} \\ \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} \end{pmatrix}$$

Implies

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & 1 & 1 & 1 \\ -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & 1 & 1 & 1 \\ -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & 1 & 1 & 1 \end{pmatrix}$$

Iff

$$\begin{pmatrix} 1 & 1 & 1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 \\ 1 & 1 & 1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 \\ 1 & 1 & 1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 \\ -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & 1 & 1 & 1 \\ -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & 1 & 1 & 1 \\ -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & -2\sqrt{\frac{1}{3}}+1 & 1 & 1 & 1 \end{pmatrix}$$

5 Further ideas

One might give the value

$$k(w,x)$$

the meaning of correlation or negative probability, but this has to be explored more in future.

It would be interesting to try to apply this point of view to SVM classification: In SVM classification one is given a semantic space $S = (X, k)$ together with True ($= +1$) and False ($= -1$) values for each x in X. The aim is to find a perspective w such that a new point $\hat{x}$ will be classified 'correctly' to True and False.

There is no meaning in SVM however to the case where some x lies on the hyperplane which distinguishes the two classes. I would argue, that here in this paper we give the meaning of 'I = indeterminate' to 0.

It would be nice to also explore the following thought:

Given a semantic space with a perspective, let us call the True elements 'axioms'. Let us call the elements of X 'atoms'. A formula should be recursively defined as: Each atom is a formula. If A, B are formulas so are $A \wedge B, A \vee B, A \to B, A \equiv B, \neg A$. We might want to count the rank / level of a formula, which should be a natural number and atoms have rank 0 and a formula $f(A, B)$ has rank $r(f(A, B)) := \max(r(A), r(B)) + 1$.

Idea: What can be proved for a set of two axioms by changing the perspective or keeping it constant? Can one show that linear indepence of the semantic vectors of a set of axioms happens if and only if the axioms are pairwise not equivalent?

6 Appendix, Sagemath / Python code

```
def kk0(a,b):
    return min(a,b)/max(a,b)

def kk1(a,b):
    return gcd(a,b)**2/(a*b)
```

```
def kk2(a,b):
    return sign(gcd(a,b))*sigma(gcd(a,b))/(sign(a)*sigma(a)*sign(b)*sigma(b))

def kkL(a,b):
    return a.dot_product(b)/(a.dot_product(a)*b.dot_product(b))

def mu(x):
    return sign(x)

def proj(w,x):
    return kk(w,x)

def And(x,y):
    return min(x,y)

def Or(x,y):
    return max(x,y)

def Implies(x,y):
    return And(1,1+y-x)

def Iff(x,y):
    return And(Implies(x,y),Implies(y,x))

def Not(x):
    return -x

def gram(rr):
    return matrix([[kk(a,b) for a in rr] for b in rr])


def Table(func,w,rr,printMu = True):
    if printMu:
        return matrix([[mu(func(proj(w,x),proj(w,y))) for x in rr] for y in rr
    else:
        return matrix([[func(proj(w,x),proj(w,y)) for x in rr] for y in rr])

def deMorgan(w,rr):
    one = all([And(Not(proj(w,x)),Not(proj(w,y)))==Not(Or(proj(w,x),proj(w,y)
    two = all([Or(Not(proj(w,x)),Not(proj(w,y)))==Not(And(proj(w,x),proj(w,y)
    return one and two

def contraposition(w,rr):
```

```
    return all([Implies(proj(w,x),proj(w,y))==Implies(Not(proj(w,y)),Not(proj

def doubleNegation(w,rr):
    return all([proj(w,x)==Not(Not(proj(w,x))) for x in rr])

def modusPonens(w,rr):
    return all([mu(Implies(And(proj(w,a),Implies(proj(w,a),proj(w,b))),proj(w
```

```
# Number logic:
#rr = [1,2,3,4,5,6]
#w = 1
#kk = kk2

# Number log-logic:
def kkl(a,b):
    x = sum([valuation(a,p)*valuation(b,p) for p in prime_divisors(gcd(a,b))])
    y1 = sqrt(sum([valuation(a,p)**2 for p in prime_divisors(a)]))
    y2 = sqrt(sum([valuation(b,p)**2 for p in prime_divisors(b)]))
    return x/(y1*y2)


rr = [1/3,1/2,2,3]
ww = [2,3]
kk = kkl

# Simplex logic:
def ee(n,N):
    return vector([1*(k==n) for k in range(1,N+1)])

N=3
rr = [ee(n,N) for n in range(1,N+1)]
v = 1/N*sum(rr)
w = v*1/v.norm()
rr.extend([-ee(n,N) for n in range(1,N+1)])

ww = [w]
kk = kkL

# Lukasiewicz logic:
e1 = vector([1,0])
e0 = vector([0,1])
```

```
rr = [-e1,e0,e1]
ww = [e0,e1,-e1]
kk = kkL


print(latex(gram(rr)))

def checks(w,rr):

    pm = False
    print("deMorgan-Rules are satisfied:")
    print(deMorgan(w,rr))

    print("contraposition satisfied:")
    print(contraposition(w,rr))

    print("doubleNegation:")
    print(doubleNegation(w,rr))

    print("modusPonens:")
    print(modusPonens(w,rr))

    print("Not")
    for x in rr:
        print(proj(w,x),Not(proj(w,x)))
        #print(mu(proj(w,x)), mu(Not(proj(w,x))))

    print("And")
    print(latex(Table(And,w,rr,printMu=pm)))

    print("Or")
    print(latex(Table(Or,w,rr,printMu=pm)))

    print("Implies")
    print(latex(Table(Implies,w,rr,printMu=pm)))

    print("Iff")
    print(latex(Table(Iff,w,rr,printMu=pm)))


for w in ww:
    print("w = ",latex(w))
    checks(w,rr)
```

```
#for n in range(1,21):
#    rr = list(range(-n,0))
#    rr.extend(range(1,n+1))
#    print(n,rr)
#    print(gram(rr).rank())
```